Okruženje i životni ciklus

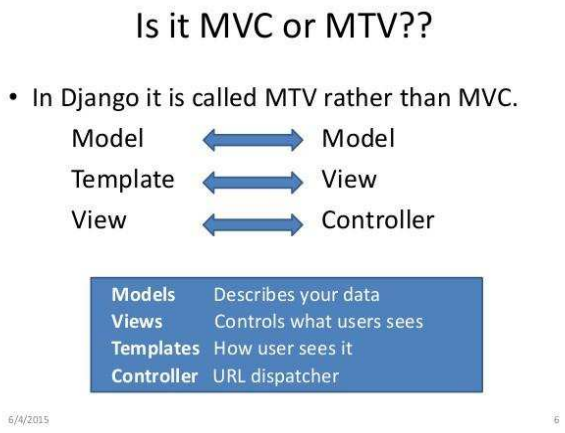
Django je Python web framework višeg nivoa namenjen brzom razvoju bezbednih sajtova lakih za održavanje. Kao takav nam i sam rešava dosta problema koji mogu nastati tokom kreiranja, konfiguracije i postavljanja sajtova, a sve to iz razloga kako bismo se mi, programeri, više usredsredili na pisanje samih aplikacija koje stoje iza sajta. Besplatan je i otvorenog koda, sa velikom bazom korisnika i dosta detaljnom dokumentacijom.

Django je prvobitno razvio tim web programera koji su radili na jednom sajtu onlajn novina između 2003. i 2005. Nakon kreiranja više sličnih sajtova, uvideli su da je dosta kodne baze bilo samo ponovo iskorišćeno. Ovaj zajednički kod koji su koristili na više sajtova evoluira u otvoren kod pod nazivom Django u julu 2005. godine.

Od prvog dana nastanka Django evoluira i napreduje kao web framework – od svoje prve verzije 1.0 u septembru 2008. do verzije 3.0 iz decembra 2019. godine. Svaka nova verzija je dodavala ispravke za poznate bagove, kao i novu funkcionalnost kao što je podrška za nove sisteme baza podataka, sisteme za šablone, keširanje, generične funkcije za preglede i klase, a sve u cilju smanjenja vremena koje je potrebno programeru da neke od tih opštih aspekata iskodira.

Sam Django je baziran na MVT šablonu (Model-Prikaz-Šablon, engl. Model-View-Template), koji se razlikuje od MVC šablona, o kojem je bilo reči ranije. Komponente MVT šablona su:

- Model – isto kao i model u MVC šablonu, odgovran je za rad sa samim podacima;
- Prikaz – kao i u kontroleru u MVC šablonu, u njemu se nalazi sva logika koju web aplikacija koristi i služi kao sprega između modela i šablona;
- Šablon – slično prikazu u MVC šablonu; odnosi se na prezentaciju podatka korisniku.



Slika 12.1. MVC naspram MTV modela[1]

[1] https://medium.com/shecodeafrica/understanding-the-mvc-pattern-in-django-edda05b9f43f

Zanimljivost

Zašto baš ime Django? Adrian Holovaty, jedan od njegovih autora, takođe je talentovan džez gitarista i veliki fan Djanga Reinhardta, jednog od najvećih i najuticajnijih džez gitarista. Reinhardt je poznat po izuzetnoj spretnosti kada je reč o gitari, iako je zbog požara u kome je izgubio mogućnost pomeranja tri prsta leve ruke mogao da svira samo kažiprstom i srednjim prstom. To je inspirisalo Adriana, koji je rekao da je kreirao Django kako bi omogućio kreiranje sajtova koristeći samo dva prsta.

Prednosti Djanga

Neke od prednosti korišćenja Djanga su:

- podrška za objektno-relaciono mapiranje – Django nam omogućava spregu između objektnog modela i najpoznatijih sistema baze podataka (MySQL, Oracle, PostgreSQL...);
- višejezična podrška – Django podržava višejezične sajtove koristeći ugrađeni sistem za internacionalizaciju, što nam omogućava kreiranje sajta koji se može prikazivati na više jezika;
- podrška za frameworkove – Django poseduje ugrađenu podršku za Ajax, RSS, keširanje i druge radne okvire;
- administracioni grafički interfejs – Django nudi gotov korisnički interfejs za administratorske akcije;
- razvojno okruženje – sam Django dolazi sa jednostavnim web serverom koji olakšava testiranje i razvoj web aplikacije.

Napomena:

Projekti i aplikacije

U tekstu do sada smo pominjali aplikacije, ali u Djangu postoje i projekti. Pod aplikacijom se ovde podrazumeva web aplikacija koja obavlja određenu aktivnost – npr. aplikacija za glasanje sa ponuđenim odgovorima, aplikacija za evidenciju događaja, baza javno dostupnih informacija itd. Projekat je kolekcija konfiguracija i aplikacija za dati sajt. Projekat može sadržati više aplikacija, dok aplikacija ne može sadržati više projekata. Zato, jedna aplikacija može biti deo i više projekata.

Kreiranje prvog Django projekta

Pre rada sa Djangom, potrebno ga je najpre instalirati komandom pip3 install django. Nakon uspešne instalacije, otvoriti komandnu liniju i pozicionirati se u folder u kojem želimo da napravimo svoj prvi projekat. Nakon što smo to učinili, treba pokrenuti komandu django-admin startproject first_project. Ovim smo u trenutnom folderu kreirali folder koji se zove first_project, što je ujedno i ime našeg prvog Django projekta. Ako bismo otvorili folder, videli bismo ovakvu strukturu:

```
first_project /
├── manage.py
├── first_project/
```

```
├── __init__.py
├── settings.py
├── urls.py
├── asgi.py
└── wsgi.p
```

Ovi fajlovi su automatski kreirani nakon pokretanja komande 'startproject'. Značenja ovih fajlova su sledeća:

- first_project/ – predstavlja direktorijum koji obuhvata naš projekat; ovo ime Djangu nije bitno i može se promeniti kasnije;
- manage.py – pomoćni skript koji nam omogućava rad sa serverom preko komandne linije; neke od komandi koje ovaj skript prima su: runserver, startapp, createsuperuser, migracije itd.;
- first_project/ – predstavlja ime direktorijuma koji će sadržati paket (kao kada bismo pravili paket za Python) koji je deo našeg projekta;
- first_project/__init__.py – prazan fajl koji govori Python prevodiocu da direktorijum u kojem se nalazi __init__.py tretira kao paket;
- first_project/settings.py – glavni konfiguracioni fajl našeg projekta; takođe, u njemu dodajemo informacije o aplikacijama koje kreiramo, tačnije u listi 'INSTALLED_APPS';
- first_project/urls.py – fajl koji definiše rute (putanje na serveru) sa prikazima;
- first_project/wsgi.py – WSGI (Web Server Gateway Interface) je posrednik između zahteva klijenta ka serveru i omogućava povezivanje korisničkog zahteva ka serverskom projektu;
- first_project/asgi.py – Asynchronous Server Gateway Interface je naslednik WSGI posrednika.

U ovom trenutku imamo projekat kreiran i dalje je potrebno kreirati prvu aplikaciju. Ovo činimo tako što se u komandnom promptu pozicioniramo u folder gde nam se nalazi manage.py fajl i pokrenemo sledeću komandu:

```
python manage.py startapp my_first_app
```

Na ovaj način smo pomoću manage.py fajla kreirali prvu aplikaciju my_first_app. Sada se u našem folderu pojavio i folder my_first_app, čija je struktura:

```
my_first_app/
├── migrations/
│       ├── __init__.py
├── admin.py
├── apps.py
├── models.py
├── test.py
└── views.py
```

Sa nekim od ovih fajlova smo se već upoznali prilikom kreiranja projekta, dok su ostali fajlovi:

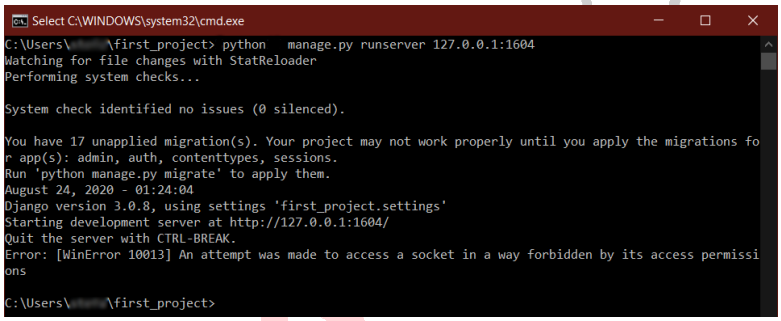
- admin.py – fajl u kojem povezujemo modele sa Django administrativnim interfejsom;
- apps.py – fajl u kojem smeštamo konfiguraciju naše aplikacije;
- models.py – fajl koji odgovara modelu naše aplikacije, zadužen za rad sa podacima;
- tests.py – fajl koji omogućava pisanje testnih slučajeva za našu aplikaciju;

- views.py – fajl u kojem pišemo logiku prikaza, koji je zapravo zadužen za rukovanje zahtevima i odgovorima.

Nakon što smo kreirali svoju prvu aplikaciju, možemo testirati uspešnost ovog procesa i uspešnost instalacije samog Djanga. Ovo činimo tako što se prethodno u komandnom promptu pozicioniramo u folder koji sadrži manage.py i pokrenemo komandu:

```
python manage.py runserver 127.0.0.1:8000
```

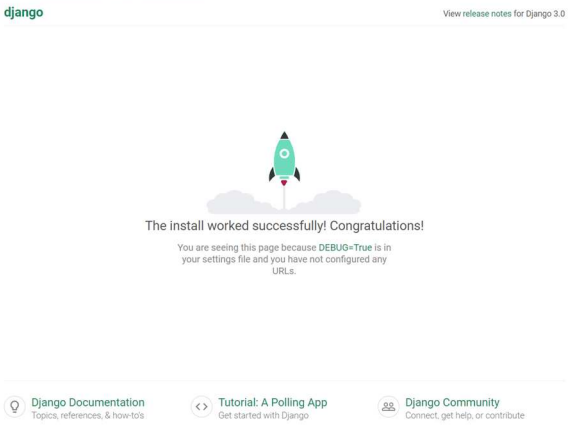
Ovoj komandi prvo prosleđujemo fajl manage.py, a njemu prosleđujemo komandu runserver kojom pokrećemo server, kao i adresu i broj porta preko koga se tom serveru može pristupiti. U našem slučaju je to lokalna IP adresa – 127.0.0.1 sa portom 8000. Ovaj broj porta se nezvanično vodi po IANA-u kao Django Development Webserver, što znači da je kao takav registrovan. Detaljnu listu registrovanih i zauzetih portova TCP protokola možemo naći na ovom linku. Ako se desi da je port zauzet, dobićemo grešku poput ove (radi namernog izazivanja greške, kako bismo posmatrali i takvu situaciju, izabrali smo već zauzeti port 1604):



Slika 12.2. Pokretanje Django projekta na zauzetom portu

U tom slučaju pokušati sa drugim portom.

Ako u pregledaču otvorimo adresu sa kojom smo prvobitno pokrenuli projekat (127.0.0.1:8000), dobićemo sledeći ekran:



Slika 12.3. Prikaz uspešne instalacije i pokretanja Djanga

Na ovaj način smo sigurno utvrdili da su instalacija Djanga, njegova konfiguracija i kreiranje prvog projekta i aplikacije uspešni.

Pitanje

Ako smo pokrenuli server sa `python manage.py runserver 127.0.0.1:8000`, da li serveru možemo pristupiti samo sa adresom 127.0.0.1?

- Da
- **Ne**

Objašnjenje:

Tačan odgovor je da ne možemo pristupiti samo adresi servera bez soketa ako smo soket iskoristili prilikom poziva komande runserver.

Django posrednici

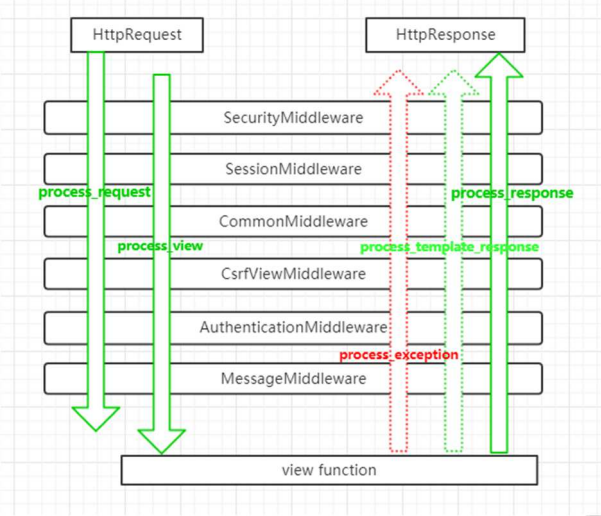
Posrednici (middlewares) se koriste još pre nego što naš prikaz dođe do toga da obradi zahtev i vrati odgovor. Lista ovih posrednika se nalazi u settings.py fajlu u poromenljivoj MIDDLEWARE tipa lista. Inicijalna lista ovih posrednika koja nastaje prilikom kreiranja samog projekta izgleda ovako:

MIDDLEWARE lista:

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Takođe je moguće da se Django projekat pokrene bez ovih posrednika, ali ako projekat kreiramo preko django-admin startproject komande, dobijamo tu, ugrađenu listu posrednika.

Svaki od tih posrednika ima različite faze i metode životnog ciklusa koje se pozivaju prilikom procesiranja zahteva. Iz toga sledi da HTTP zahtev mora proći sve posrednike i da se mora istom tom putanjom vratiti kao odgovor. Na primer, pošto smo rekli da svaki od posrednika obavlja određenu funkciju, posrednik iz naše liste, AuthenticationMiddleware, zadužen je za povezivanje klijenata sa njihovim zahtevima koristeći sesije. Takođe je bitno da Django posrednike iz liste MIDDLEWARE primenjuje odozgo nadole kada je reč o dolazećem zahtevu i da ih ponovo primenjuje redom, ali odozdo nagore, kada je reč o odgovoru, što se može videti na sledećoj slici:



Slika 12.4. Kretanje zahteva i odgovora kroz posrednike[2]

Slojevi Django aplikacije i životni vek zahtev–odgovor toka

Postoji šest slojeva iz kojih se sastoji jedna Django aplikacija:

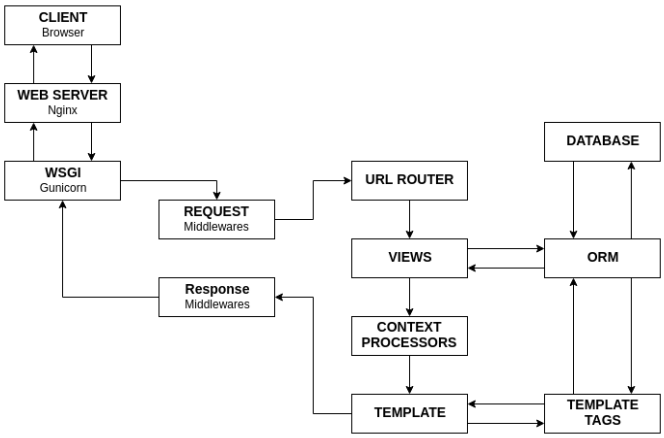
1. posrednici (middleware) zahteva;
2. URL ruteri;
3. prikazi;
4. procesori konteksta;
5. vizuelizovanje šablona;
6. posrednici odgovora.

Po pristizanju HTTP zahteva nastupaju i posrednici zahteva i oni se primenjuju po redosledu po kojem su sami zahtevi i došli. Nakon ovog procesiranja, zahtev se šalje URL ruteru, koji preuzima putanju iz samog zahteva i pokušava da upari putanju koja dolazi sa zahtevom sa raspoloživim putanjama na serveru koje su definisane u urls.py fajlu. Ako je pronađena odgovarajuća putanja, zahtev se dalje prosleđuje prikazu (views.py) koji odgovara ovoj putanji.

Prikazi sada koriste podatke iz poslatog zahteva kako bi ih procesirali i generisali potrebne povratne podatke (kontekst), koji će pomoću šablona zapravo generisati odgovor (vizuelizacija šablona) koji će se poslati nazad klijentu.

Opet, sada ovaj odgovor mora proći kroz iste posrednike kroz koje je prošao i na početku pre nego što ga klijent primi. Posrednici odgovora kroz koje odgovor prolazi mogu ga naknadno izmeniti tako što će mu dodati/izmeniti polja zaglavlja.

[2] https://developpaper.com/details-of-django-middleware/



Slika 12.5. Kretanje zahteva i odgovora kroz Django radni okvir[3]

Rezime

- Django je Python web framework višeg nivoa namenjen brzom razvoju bezbednih i lako održavanih sajtova.
- Django je prvobitno razvio, između 2003. i 2005. godine, tim web programera koji su radili na jednom sajtu onlajn novina.
- Sam Django je baziran na MVT šablonu (Model-Prikaz-Šablon, engl. Model-View-Template).
- Projekat može sadržati više aplikacija, dok aplikacija ne može sadržati više projekata; zato, jedna aplikacija može biti deo i više projekata.
- Pre upotrebe Djanga potrebno je instalirati ga komandom pip3 install django.
- Za kreiranje Django projekta pozivamo: django-admin startproject first_project.
- Za kreiranje aplikacije u okviru kreiranog Django projekta pozivamo: python manage.py startapp my_first_app.
- Posrednici (middlewares) se koriste još pre nego što naš prikaz dođe do toga da obradi zahtev i vrati odgovor.
- Django aplikacija sastoji se iz šest slojeva:
    - posrednici (middleware) zahteva;
    - URL ruteri;
    - prikazi;
    - procesori konteksta;
    - vizuelizovanje šablona;
    - posrednici odgovora.

[3] https://medium.com/@goutomroy/request-and-response-cycle-in-django-338518096640